

Программирование Python

Графика

Кафедра ИВТ и ПМ
ЗабГУ

2018

План

Работа с изображениями. Пакет Pillow

- Введение

- Изображение как массив `numpy.array`

- Дополнительно. Фильтры

- Рисование графических примитивов

Анимация и интерактивная графика

- `graphics.py`

- `pygame`

- Другие библиотеки

Ссылки и литература

Outline

Работа с изображениями. Пакет Pillow

Введение

Изображение как массив `numpy.array`

Дополнительно. Фильтры

Рисование графических примитивов

Анимация и интерактивная графика

`graphics.py`

`pygame`

Другие библиотеки

Ссылки и литература

Outline

Работа с изображениями. Пакет Pillow

Введение

Изображение как массив `numpy.array`

Дополнительно. Фильтры

Рисование графических примитивов

Анимация и интерактивная графика

`graphics.py`

`pygame`

Другие библиотеки

Ссылки и литература

Для Python 2 был создан пакет для работы с графикой PIL (Python Imaging Library). Однако в 2011 году разработка пакета была прекращена.

После прекращения разработки старый пакет получил новое развитие с названием Pillow.

Однако для обратной совместимости имя используемое для подключения пакета осталось прежним: PIL

Documentation

Установка

```
pip install Pillow
```

Image

Для работы с изображениями используется специальный класс **Image**, который содержится в модуле *Image*.

`PIL.Image.Image`

С помощью этого класса можно создавать в памяти новые изображения ("холсты") или загружать их из файла.

Image

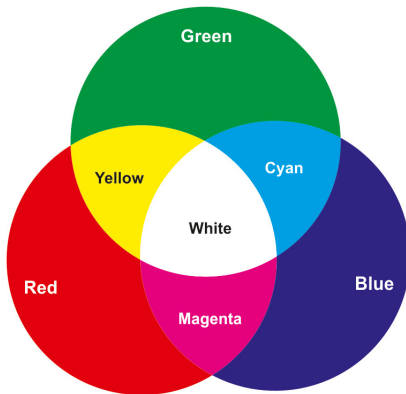
Функции модуля Image

- ▶ **new(mode, size) -> Image** Создаёт изображение
mode - строка - задаёт способ представления изображения
size - кортеж: (ширина, высота)
размеры указываются в пикселях
- ▶ **open(filename) -> Image**
Загружает изображение из файла
filename - имя файла

Способы кодирования цвета

Чаще всего изображение представляют как набор киселей с цветом кодируемым интенсивностью трех цветов: красным, зелёным и синим.

RGB = red, green, blue.



RGB - Red, Green, Blue

© Gil Dekel PoeticMind.co.uk

Способы кодирования цвета

Возможные значения параметра mode

- ▶ 1 (1-bit pixels, black and white, stored with one pixel per byte)
- ▶ L (8-bit pixels, black and white)
- ▶ P (8-bit pixels, mapped to any other mode using a color palette)
- ▶ RGB (3x8-bit pixels, true color)
- ▶ RGBA (4x8-bit pixels, true color with transparency mask)
- ▶ CMYK (4x8-bit pixels, color separation)
- ▶ YCbCr (3x8-bit pixels, color video format)
- ▶ LAB (3x8-bit pixels, the L^*a^*b color space)
- ▶ HSV (3x8-bit pixels, Hue, Saturation, Value color space)
- ▶ I (32-bit signed integer pixels)
- ▶ F (32-bit floating point pixels)

Интенсивность отдельного цвета задаётся числом. Как правило это число от 0 до 255, для хранения которого отводится один байт.

Белый - (255, 255, 255)

...

Серый - (128, 128, 128)

...

Чёрный - (0, 0, 0)

Таким образом чтобы задать цвет одного пикселя используются три числа, то есть три байта.

Тогда изображение будет представлять собой матрицу, каждый элемент которой - тройка чисел кодирующих цвет.

rapidtables.com/web/color/RGB_color.html - цвета и соответствующие им коды.

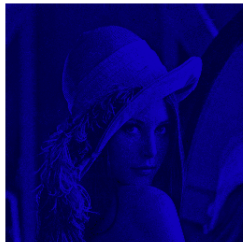
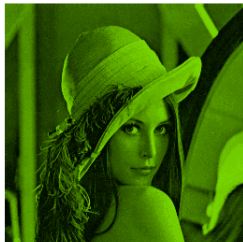
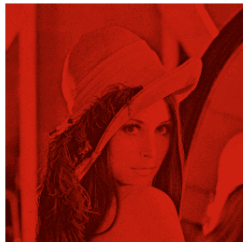


Image. Создание или загрузка

```
from PIL import Image, ImageDraw

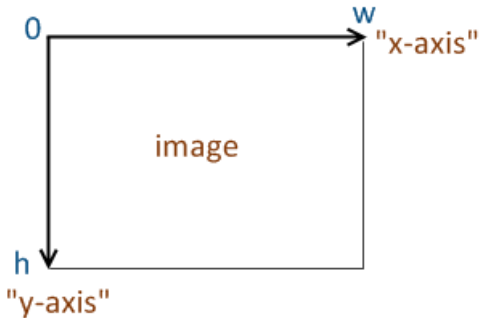
# создание RGB изображения размером 200 x 200 пикселей
img1 = Image.new("RGB", (200, 200))

# загрузка изображения из файла
ima2 = Image.open('/path/to/photos/jelly.jpg')
```

Система координат изображения

Начало координат расположено в левом верхнем углу изображения.

Положительное направление оси y - вниз.



Пример 1

```
from PIL import Image, ImageDraw
```

```
# ширина и высота изображения в пикселях
```

```
W,H = 32, 32
```

```
# создание RGB изображения
```

```
img = Image.new("RGB", (W, H))
```

```
for i in range(H):
```

```
    # изменение цвета пикселя с координатами i,i
```

```
    # на цвет заданный как (R,G,B)
```

```
    img.putpixel( (i,i), (160,128,64) )
```

```
# сохранить изображение в файл
```

```
# формат изображения определяется по расширению файла
```

```
img.save("example1.png")
```

Пример 1.1

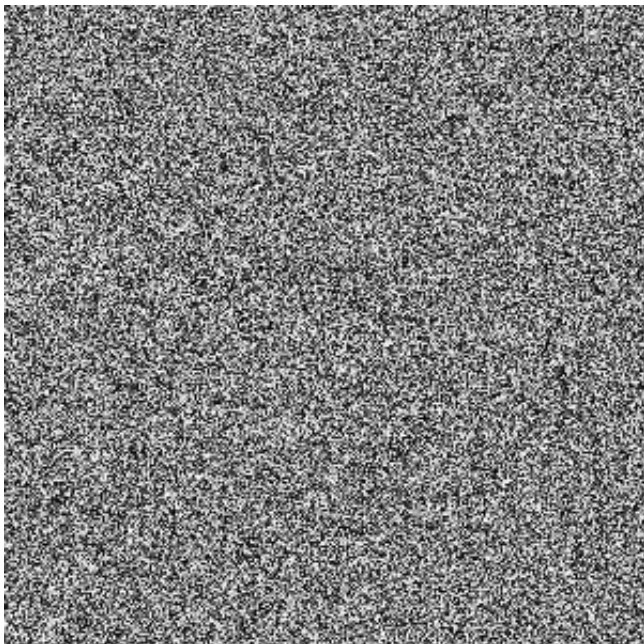
```
from PIL import Image, ImageDraw
from random import randint
# ширина и высота изображения в пикселях
W,H = 300, 300
# создание RGB изображения
img = Image.new("RGB", (W, H))

for i in range(W):
    for j in range(H):
        # изменение цвета пикселя с координатами i,i
        # на цвет заданный как (R,G,B)
        c = randint(0,255)
        img.putpixel((i,j),(c,c,c))

# сохранить изображение в файл
# формат изображения определяется по расширению файла
img.show()
img.save("example1.png")
```

Как будет выглядеть изображение?

Пример 1.1



Некоторые методы класса Image

► Изменение размера¹

`Image.resize(size, resample=0) -> Image`

size - кортеж с новыми размерами

resample - обозначает алгоритм интерполяции

► Обрезка изображения

`Image.crop(box=None) -> Image`

box - кортеж с координатами прямоугольника (левый верхний угол, правый нижний)

► Повернуть изображение

`Image.rotate(angle, resample=0, expand=0)`

Поворачивает изображение на указанное число градусов (angle) по часовой стрелке.

Не поместившиеся в рамки изображения края отсекаются.

pillow.readthedocs.io/en/3.1.x/reference/Image.html - документация Image

¹habr.com/post/243285 - Ликбез: методы ресайза изображений

Outline

Работа с изображениями. Пакет Pillow

Введение

Изображение как массив `numpy.array`

Дополнительно. Фильтры

Рисование графических примитивов

Анимация и интерактивная графика

`graphics.py`

`pygame`

Другие библиотеки

Ссылки и литература

Изображение как массив numpy.array

Для математической обработки изображения его переводят в двумерный массив **array** из популярной математической библиотеки **numpy**.

```
from PIL import Image
import numpy as np

img = Image.open("example2.png")

# Image -> array
img_mat = np.asarray(img)

img_mat.shape # (300, 300, 3)

# обратное преобразование: array -> Image
img2 = Image.fromarray(img_mat)
img2.show()
```

Картинка будет представлена как массив размерностью $300 \times 300 \times 3$. Где последний индекс будет соответствовать цветовой компоненте пикселя.

Изображение как массив numpy.array

```
img = Image.open("example2.png")
img_mat = np.asarray(img)

# доступ к пикселю из Image
img.getpixel( (5, 20) )
# (191, 64, 64)

# доступ к пикселю из numpy.array
# здесь первой указывается координата Y (номер строки матрицы)
img_mat.getpizel( (20, 5) )
# [191, 64, 64]
```

В примере использована картинка со слайда 33

Загрузка изображений как numpy.array

С помощью пакета matplotlib можно загружать изображения можно сразу представляя их как многомерный массив array.

```
import numpy as np
import matplotlib.pyplot as plt
```

```
img_mat = plt.imread('example2.png')
```

```
#показать изображение
```

```
plt.imshow(img_mat)
```

```
plt.axis('off') # отключить координатные оси
```

Outline

Работа с изображениями. Пакет Pillow

Введение

Изображение как массив `numpy.array`

Дополнительно. Фильтры

Рисование графических примитивов

Анимация и интерактивная графика

`graphics.py`

`pygame`

Другие библиотеки

Ссылки и литература

Графические фильтры

На основе попиксельной обработки изображения построены **графические фильтры**.

Графические фильтры применяются например для размытия изображений, увеличения чёткости или выделения контуров.

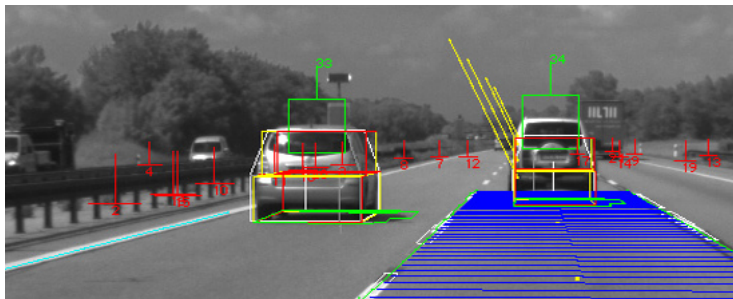
- ▶ Медианный фильтр
- ▶ habrahabr.ru/post/142818 - Матричные фильтры обработки изображений
- ▶ habrahabr.ru/company/gilalgorithms/blog/67594/ - Очистка изображения от шума, некоторые методы
- ▶ www.intuit.ru/studies/courses/993/163/lecture/4505 - Фильтрация изображений

Примеры фильтров



Компьютерное зрение

Графические фильтры используются и для облегчения восприятия изображения или для выделения важных свойств изображения для последующего применения алгоритмов компьютерного зрения.





OpenCV - Open Source Computer Vision Library) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом.

Outline

Работа с изображениями. Пакет Pillow

- Введение

- Изображение как массив `numpy.array`

- Дополнительно. Фильтры

- Рисование графических примитивов

Анимация и интерактивная графика

- `graphics.py`

- `pygame`

- Другие библиотеки

Ссылки и литература

Класс Draw

Для рисования графических примитивов используется отдельный класс **Draw** из модуля ImageDraw.

```
from PIL import ImageDraw
```

pillow.readthedocs.io/en/3.1.x/reference/ImageDraw.html -
Документация по модулю ImageDraw

Задание координат

При изображении графических примитивов, например эллипса вместо координат центра указываются координаты прямоугольника, куда фигура должна быть вписана. Обычно формальный параметр принимающий эти координаты называется xu .

Эти координаты представляются списком из двух кортежей $[(x1, y1), (x2, y2)]$

$(x1, y1)$ - координаты левого верхнего угла прямоугольника;
 $(x2, y2)$ - координаты правого нижнего угла прямоугольника.
Формальный параметр задающий такой прямоугольник как правило называется xu .

Задание цвета

Для RGB изображений цвет задаётся кортежем из трёх чисел от 0 до 255 соответствующих интенсивностям красного, зелёного и синего.

Для незамкнутых контуров, например линии цвет задёт формальный параметр **fill**.

Для замкнутых, например прямоугольника, формальный параметр **fill** задаёт заливку, а параметр **outline** - цвет контура.

Некоторые методы

- ▶ Точка

```
PIL.ImageDraw.Draw.point(xy, fill=None)
```

- ▶ Прямая

```
PIL.ImageDraw.Draw.line(xy, fill=None, width=0)
```

- ▶ Эллипс(окружность)

```
PIL.ImageDraw.Draw.ellipse(xy, fill=None, outline=None)
```

- ▶ Прямоугольник

```
PIL.ImageDraw.Draw.rectangle(xy, fill=None, outline=None)
```

- ▶ Текст

```
PIL.ImageDraw.Draw.text(xy, text, fill=None,  
                        font=None, anchor=None)
```

Пример 2

```
from PIL import Image, ImageDraw
```

```
W,H = 256, 256
```

```
img = Image.new("RGB", (W, H))
```

```
# создание "рисовальщика"
```

```
draw = ImageDraw.Draw(img)
```

```
for i in range(0, H+1, 8):
```

```
    # цвет пикселя зависит от переменной цикла
```

```
    draw.line([(0, 0), (W,i)], fill=(i,255-i,i), width=2)
```

```
    draw.line([(0, 0), (i,H)], fill=(255-i,i,i), width=2)
```

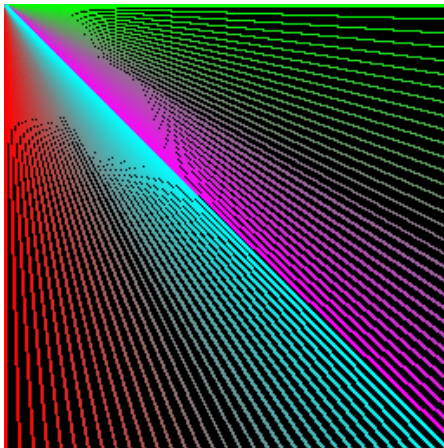
```
# удаление "рисовальщика"
```

```
del draw
```

```
img.save("example2.png")
```

```
img.show()
```


Пример 2



Общие рекомендации

Во время создания изображений из графических примитивов, рекомендуется процедурный подход.

В случаях когда можно избавиться от использование жёстко заданных координат, следует это сделать.

Другие пакеты



Пакет **scikit-image** содержит алгоритмы для работы с изображениями: фильтрация, геометрические преобразования, feature detection и т.д.

Outline

Работа с изображениями. Пакет Pillow

- Введение

- Изображение как массив `numpy.array`

- Дополнительно. Фильтры

- Рисование графических примитивов

Анимация и интерактивная графика

- `graphics.py`

- `pygame`

- Другие библиотеки

Ссылки и литература

Outline

Работа с изображениями. Пакет Pillow

- Введение

- Изображение как массив `numpy.array`

- Дополнительно. Фильтры

- Рисование графических примитивов

Анимация и интерактивная графика

- `graphics.py`

- `pygame`

- Другие библиотеки

Ссылки и литература

graphics.py - модуль для простой анимированой и интерактивной графики.

Не входит в стандартную библиотеку python

Установка:

```
pip install graphics.py
```

Статичная графика

```
from graphics import *  
  
win = GraphWin("My Circle", 100, 100)  
  
c = Circle(Point(50,50), 10)  
c.draw(win)  
win.getMouse() # Pause to view result  
win.close()    # Close window when done
```

Простая анимация

```
from graphics import *
from time import sleep

win = GraphWin("My Circle", 100, 100)

i = 0
while True:
    # удалим все объекты из окна
    for item in win.items:
        item.undraw()
    # создадим новую окружность
    c = Circle(Point(50,50), i)
    # нарисуем окружность на окне
    c.draw(win)
    # приостановим программу на 1/30 секунды
    sleep(1/30)
    i = (i+1) % 50
```


Outline

Работа с изображениями. Пакет Pillow

- Введение

- Изображение как массив `numpy.array`

- Дополнительно. Фильтры

- Рисование графических примитивов

Анимация и интерактивная графика

- `graphics.py`

- `pygame`**

- Другие библиотеки

Ссылки и литература

Фреймворк (framework — остов, каркас, структура) — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

pygame

pygame - модуль (фреймворк) для создания игр на Python.

Документация pygame

- ▶ Основные функции для работы с отображением
pygame.org/docs/ref/display.html
- ▶ Клавиши - pygame.org/docs/ref/key.html
- ▶ Рисование - pygame.org/docs/ref/draw.html

Outline

Работа с изображениями. Пакет Pillow

- Введение

- Изображение как массив `numpy.array`

- Дополнительно. Фильтры

- Рисование графических примитивов

Анимация и интерактивная графика

- `graphics.py`

- `pygame`

- Другие библиотеки

Ссылки и литература

Другие библиотеки

PyQtGraph используется для создание интерфейса пользователя и в том числе для отображения 2D и 3D графики

<http://www.pyqtgraph.org/documentation/introduction.html>

Outline

Работа с изображениями. Пакет Pillow

- Введение

- Изображение как массив `numpy.array`

- Дополнительно. Фильтры

- Рисование графических примитивов

Анимация и интерактивная графика

- `graphics.py`

- `pygame`

- Другие библиотеки

Ссылки и литература

Ссылки и литература

- ▶ habrahabr.ru/post/163663 - Играемся с изображениями в Python
- ▶ stepik.org/course/1280/ - Обработка изображений (youtube)
- ▶ myshared.ru/slide/309204/ - Обработка изображений Image processing. - презентация
- ▶ hscipy-lectures.org/advanced/image_processing/ - **Image manipulation and processing using Numpy and Scipy**
- ▶ Документация и примеры использования pygame (англ) pygame.org/docs
- ▶ gamedev.ru

Ссылки и литература

Ссылка на слайды

github.com/VetrovSV/Programming